Outline

- Extensions.
- Concurrent optimization and binding.
  - Binding and polarity assignment.
  - Boolean covering/matching.
  - Structural covering/matching.
- Algorithms for library binding.
  - Rule-based systems for library binding.
- Modeling and problem analysis.

---

Stanford University

Giovanni De Micheli

LIBRARY BINDING

GDM
Given an unbound logic network

Library binding

– Transform network into an interconnection of instances of library cells.
– Method used for re-designing circuits in different technologies.

Called also technology mapping:
– Optimise area, possible under delay constraints.
– Optimise delay, possible under area constraints.

Miscellaneous:
– Schmitt triggers.

Library models

– Single-output functions: e.g. AND, OR, AOI.
– Compound cells: e.g. adders, encoders.

Sequential elements:
– Registers, counters.

Combinational elements:
– Compound models: e.g. AND, OR, AOI.
– Single-output functions: e.g. AND, OR, AOI.

– Miscellaneous:

– Schmitt triggers.
Rule-based systems:
- Mimic designer activity.
- Handle all types of cells.

Heuristic algorithms:
- Restricted to single-output combinational cells.
- Handle all types of cells.
- Mimic designer activity.

Most tools use a combination of both.

Major approaches:

- Handle high-fanout problems, buffering, etc.
- Select subnetwork to be mapped.

Rules:

Implementation:
- Set of patterns associated with best

Data-base:
- Binding by stepwise transformations.

Rule-based library binding

Binding by stepwise transformations.

Rule-based systems binding
Meta-rules determine dynamically breadth and depth.

- Depth (look-ahead).
- Breadth (options at each step).

Search space:

- Search for a sequence of transformations.
Rule-based library binding

Advantages:

– Applicable to all kinds of libraries.

Disadvantages:

– Larger rule data-base.

Algorithms for library binding

or equivalent pattern.

– Each cell modeled by its function.

Library description/update is straightforward.

– Quality comparable to rule-based systems.

– Fast and efficient.

Restricted to S-outputs combinational cells.

GDM

Formal properties of bound network.

Completeness issue.

Data-base updates.

Library description/update is straightforward.

– Quality comparable to rule-based systems.

– Fast and efficient.

Restricted to S-output combinational cells.

GDM

Advantages:

– Applicable to all kinds of libraries.

Rule-based library binding
Problem analysis

Assumptions

- Replacement of each vertex by base cell.
- Trivial binding.
- Example: 2-input AND, OR, NAND, NOT.
- Decomposition into simple base functions.
- Network granularity is fine.

Matching:
- A cell matches a sub-network, if their terminal behavior is the same.
- A cover of an unbound network is a partition into subnetworks, which can be replaced by library cells.

Covering:
- A cover of an unbound network

Assumptions

- Network granularity is fine.
- Decomposition into simple base functions.
- Example: 2-input AND, OR, NAND, NOT.
- Replacement of each vertex by base cell.
- Trivial binding.
- Example: 2-input AND, OR, NAND, NOT.
Heuristic Algorithms

Example
Heuristic algorithms

- More powerful
- Solve tautology problem
- Use Boolean models

Boolean approach:
- Rely on pattern matching techniques
- Example: trees, dag's
- Model functions by patterns

Structural approach:
-
Patterns do not match.

\[ zx + \beta x + \hat{x} = \hat{b} \]

\[ z \beta + \beta x + \hat{x} = \hat{f} \]

Boolean versus structural matching

Patterns may be different:

Boolean match.

Functions equality is a tautology:

\[ zx + \beta x + \hat{x} = \hat{b} \]

\[ z \beta + \beta x + \hat{x} = \hat{f} \]
Structural matching and covering

Expression patterns:
- Represented by dags.

Identification patterns in network:
- Sub-graph isomorphism.

Simplification:
- Use tree patterns.

Example:
\[ \begin{array}{c}
(\text{c}) \\
\end{array} \]
Tree-based matching

- Aho-Corasick automaton.
- Simple binary tree match.
- Pattern recognition.
- Possibly more than one tree per cell.
- Represented by trees.

Library:

- Subject tree.
  * General base functions.
  * NOR2 (or NAND2) + INV.

Network:

- Partitioned and decomposed.

Simple library

(a) (b) (c) (d) (e) (f) (g)
Binary tree match

Example: match

Decompose into.

- Consider INV:
  - Number of children mismatch.
- Consider NAND2:
  - Same number of children.

Recursive call:

\[
(q_i, v) = pf
\]

De-compose into

\[
\]

- Leaves found: MATCH!

Recursive call:

\[
\]

Example: match

Binary tree match
Treecovering

Example

Dynamic programming:

- Optimum solution, for the subtree.
- All library cells.
- Locally rooted subtree.
  - Attempt to match:
    - At each vertex:
      - At visit subject tree bottom-up.


cost = 2
cost = 3
cost = 4
cost = 5
The library.

- Cannot happen when base functions are in

  There is no match:

  - Plus the labels of the vertices.

  - The vertex is labeled with the cell cost.

  - The cell tree is isomorphic to a subtree with

  - The vertex is labeled with the cell cost.

  - The cell pattern tree and the rooted subtree are

  - The cell pattern tree and the rooted subtree are

  - The cell pattern tree and the rooted subtree are

Tree covering
Example

A0121 led by a NAND2 gate.

- Best choice:
  - INV: 2; NAND2: 3; AND2: 4; AOI21: 6.

- Area costs:

Minimum-area cover:

<table>
<thead>
<tr>
<th>Gate</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV</td>
<td>2</td>
</tr>
<tr>
<td>NAND2</td>
<td>3</td>
</tr>
<tr>
<td>AND2</td>
<td>4</td>
</tr>
<tr>
<td>AOI21</td>
<td>6</td>
</tr>
</tbody>
</table>

Best choice:

AOI21 fed by a NAND2 gate.
* Binning techniques.
* Load fanout unknown.
  - Load-dependent delay.
  * Straightforward.
  - Constant gate delay.

Delay modeling:

Cost related to gate delay:

Dynamic programming approach:

---

Minimum delay cover

---

Example

---

Minimum delay cover

---

Example
Minimum delay cover
constant delays

The cell pattern tree and the rooted subtree are isomorphic.

- The vertex is labeled with the cell delay.
- The cell tree is isomorphic to a subtree with leaves $I$.

Example

- AND2: two NAND2 and an INV gate.

Best choice:
- Compute data-ready times bottom-up:
- INV: 2; NAND2: 4; AND2: 5; AOI21: 10.

Constant delays:
- except for $t_p = 6$;
  - Inputs data-ready times are 0

Minimum delay cover

- plus the maximum of the labels of $I$.
  - The vertex is labeled with the cell cost
    - ship $I$.  
  - The cell is labeled with the cell delay.
  - Isomorphic:
    - The cell pattern tree and the rooted subtree are isomorphic.
Minimum delay cover

Load-dependent delays

Example
Example

• Solution uses SINV cell.
  • Assume output load is 5.
• Same solution as before.
  • Assume output load is 1.

• Load-dependent delays:
  • INV: 1 + l; NAND2: 3 + l; AND2: 4 + l; AOI21: 9 + l.

• Loads:
  • INV: 1; NAND2: 1; AND2: 1; AOI21: 1.

• Assume output load is 1:
  • SINV: 1 + 0.5 l.

• Assume output load is 5:
  • Solution uses SINV cell.

• Load-dependent delays:
  • INV: 1 + l; NAND2: 3 + l; AND2: 4 + l; AOI21: 9 + 1.

• Loads:
  • INV: 1; NAND2: 1; AND2: 4; AOI21: 9 + 1.

• Except for $t_p = 6$:
  • Inputs data-ready times are 0.

Example
$\text{Subject graph}$

$\text{\textbf{Boolean covering}}$

- Limit size and depth of clusters.
- Several functions associated with \( v_i \):
  - Construct clusters by local elimination.
  - When constructing vertex \( v_i \):
    - Decompose network into base functions.

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Match</th>
<th>Load=1</th>
<th>Load=2</th>
<th>Load=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example
Boolean matching

over all permutations of input variables.

- Tautology check must be done

such that \( f(x) \) is a tautology?

- Exists a permutation matrix \( P \)

  - Tautology check:

    - Sub-network behavior:

      - Cluster function \( f(x) \)

    - Cell behavior:

      - Pattern function \( g(y) \)

\[ (p + q)(p + r + s) = \begin{cases} \text{false} & \text{if} \; p = q \\ \text{true} & \text{otherwise} \end{cases} \]

\[ \bar{h}(p + r + s) = \begin{cases} \text{false} & \text{if} \; p = q \\ \text{true} & \text{otherwise} \end{cases} \]

\[ (p + q)(q + r) = \begin{cases} \text{false} & \text{if} \; p = q \\ \text{true} & \text{otherwise} \end{cases} \]

\[ \bar{h}(p + q) = \begin{cases} \text{false} & \text{if} \; p = q \\ \text{true} & \text{otherwise} \end{cases} \]

\[ \bar{h}x = \begin{cases} \text{false} & \text{if} \; p = q \\ \text{true} & \text{otherwise} \end{cases} \]
Boolean matching

Any input permutation must associate
- unate (binate) variables in $P$ with unate (binate) variables in $S$.

Variables or groups of variables
- that are interchangeable in $(\lambda) q = f(x)$
- must be interchangeable in $f(x)$.

That are interchangeable in $f(x)$
- with unate (binate) variables in $P$.

Any input permutation must associate
- unate (binate) variables in $f(x)$.

Example

Cluster function:

Pattern functions:

Symmetries:

Symmetries:

Symmetries:
Search for lower cost solution by not constraining the signal polarities.

Most circuits allow us to choose the input/output signal polarities.

Approaches:

- Boolean covering.
- Structural covering.

Structural covering and polarity assignment

Pre-process subject network:
- Add inverter-pair cell to the network.
  - To eliminate unneeded pairs.
  - Add inverter-pair cell to the network.
  - Provide signals with both polarity.
  - Add inverter pairs between NANDs.

Pre-process subject network:

Library binding

and polarity assignment

Structural covering

Polarity assignment

and polarity assignment

Structural covering
Boolean covering

and polarity assignment

Example
Concurrent optimization and library binding

Motivation:

- Logic simplification is usually done prior to binding.
- Binding induces some don't care conditions.
- Boolean covering/matching can exploit don't care conditions.

Mechanism:

- Logic simplification/substitution can be combined with binding.
- Boolean covering/matching can exploit don't care conditions.

Match pattern function $g(x,y) = \bar{h} + x = (\bar{h}, x, y)$ in the set:

$\{q, p, q, p, q, v, q, q + p, q + v, q + p, q + v\}$
Example

Concurrent logic optimization and binding

- Lower cost solutions.
- Library binding and optimization.

Concurrent approach:

- Bind cells after simplification.
- Use don't care information for simplification.

Standard approach:

- Library binding and optimization.
- Use don't care information for simplification.
Matching with DC. Mapping into MVX gate.

- No simplification. Mapping into AOI gate.
- Consider $\rho, \rho x = \rho (\rho + a)x = f x$
- Don't care set includes $x$.
- Assume $v, x$ is bounded to $O(R3\{r, b\})$.

Example

![Diagram 1]

Example

![Diagram 2]
Matching compatibility graph

Vertices:
- Boolean cover (or the error).
- Classes of functions.

Edges:
- Representative functions: edges differ in one minterm.
- N-PM classes of functions.

Path:
- Boolean cover (or the error).

Example
Given cluster function \( f \). 

- Find representative function \( \rho \). 
- For all paths leading to a library element: 
  - Check if error is contained in local don't care set. 

Example

\( \rho \rho + b \epsilon \) and thus can match \( \epsilon \).

Representative function \( \rho \rho + b \epsilon \) is in the same \( N \). 

\( \rho \rho + b \epsilon \) 

- Multiplexer gate: 
  - Only vertex \( v_9 \) is a library cell.
  - Because error included in the don't care set.
  - \{ \( 3, 4, 6, 7 \) \} 
  - Vertices reachable from \( v_9 \): 
    - Representative vertex \( v_9 \). 
  - Cluster function: \( x = f + a + c \) with CDC. 

- \( \rho \rho \).
Library binding is very important.

Rule-based approach:

- Boolean: slower but promising.
- Pattern-based: fast but limited.

Algorithmic approach:

- General, sometimes inefficient.

Summary